On Critical Orientations in the Kedem-Sharir Motion Planning Algorithm for a Convex Polygon in the Plane

Klara Kedem, Micha Sharir and Sivan Toledo
Department of Computer Science, Tel-Aviv University, Tel-Aviv, Israel

Abstract

We discuss a technical problem arising in the motion planning algorithm of Kedem and Sharir [KS], and propose a way to overcome it without increasing the asymptotic complexity of the algorithm.

1 Introduction

The paper "An efficient motion-planning algorithm for a convex polygonal object in two-dimensional polygonal space", by Kedem and Sharir [KS], studies the problem of planning a collision-free motion (including translation and rotation) for a convex polygonal body B, with k corners, amidst polygonal obstacles having n corners altogether. Given initial and final placements of B, determine whether there is an obstacle-avoiding motion from the initial placement to the final placement and, if so, plan such a motion.

In what follows we assume some familiarity of the reader with the algorithm of [KS]. Nevertheless we will present a brief description of the technique, providing enough details to allow us to state the technical difficulty that arises, and to describe a method for overcoming that difficulty.

The first author was supported by the Eshkol grant 04601-90 from the Israeli Ministry of Science and Technology. The second author was partly supported by the Fund for Basic Research administered by the Israeli Academy of Sciences, by National Science Foundation Grant CCR-91-22103, and by grants from the U.S.-Israeli Binational Science Foundation, and the G.I.F., the German-Israeli Foundation for Scientific Research and Development. The third author was partly supported by the Interdisciplinary Program at Tel-Aviv University. First author's current address: Department of Mathematics and Computer Science, Ben Gurion University of the Negev, Beer Sheva, Israel. Third author's current address: Department of Computer Science, MIT, Boston, Ma.

1.1 Outline of the Kedem-Sharir Algorithm

The technique of [KS] is to construct a combinatorial representation of the boundary of the three-dimensional space FP of free placements of B, parametrized by (x, y, θ), where (x, y) are the coordinates of some fixed reference point on B, and θ is the orientation of B. The representation of FP is by means of a so-called *edge graph* EG, whose nodes are edges of FP, and whose edges connect nodes in EG if the corresponding edges of FP are adjacent in some cross-section (at a fixed θ) of FP. The edge graph preserves the connectivity of FP—each connected component of EG consists of all edges that bound the same connected component of FP. Once the edge graph is constructed, motion planning is reduced to path searching in that graph.

The edge graph EG is built incrementally, by sweeping a planar cross-section of the form $\theta = const$ over the space FP. We denote by FP_θ the cross-section of FP at a fixed orientation θ, hence it represents the space of free placements of B when B is allowed only to translate at the fixed orientation θ. As shown e.g. in [KLPS], the space FP_θ is a planar polygonal region that has only $O(kn)$ vertices and edges. In more detail, let B_θ denote the standard placement of B at which it has orientation θ and its reference point lies at the origin. Assume, without loss of generality, that the obstacles are a collection of convex polygons with pairwise disjoint interiors, and let us denote them by $A_1, \ldots, A_m$. Let $A_i^*(\theta)$ denote the *Minkowski sum* $A_i + (-B_\theta)$, for $i = 1, \ldots, m$. Then FP_θ is the complement of the union of the *expanded obstacles* $A_i^*(\theta)$. It is shown in [KLPS] that the boundaries of any pair of distinct expanded obstacles cross in at most two points, and that this implies that the union of the expanded obstacles has at most $6m = O(n)$ concave boundary corners. Hence FP_θ has at most $O(n)$ convex corners and at most $O(kn)$ concave corners (which are corners of the expanded obstacles $A_i^*(\theta)$). Note that each convex vertex of FP_θ represents a placement of B at orientation θ at which it makes simultaneously two distinct contacts with the obstacles, while otherwise remaining free of collision with the obstacles;

each concave corner of FP_θ represents a placement of B at which a corner of B makes contact with some obstacle corner, while otherwise remaining free.

The construction of EG begins by constructing a *vertex graph* VG_{θ_0} at the orientation θ_0 of the initial placement of B. This graph represents the boundary of FP_{θ_0} when B is allowed to translate at the fixed initial orientation θ_0—its nodes are the vertices of FP_{θ_0} and its edges connect pairs of adjacent vertices along the boundary of FP_{θ_0} (see [KS] for a more precise formulation). As θ varies, the vertices of VG_θ, as they trace the edges of FP, are followed. Sweeping through FP in this manner, we need to detect *critical orientations*, at which edges of FP start or end. Roughly speaking, a critical orientation is an orientation of B at which there exists a placement where B makes three simultaneous contacts with the obstacles, while otherwise remaining free. Once the list of all critical orientations is constructed (and sorted), the sweep over FP can be performed, because the critical events that occur during the sweep are already known and the graphs VG_θ and EG can be updated at each critical orientation. Except for various additional technical details, which we omit here, this is a high-level description of the algorithm of [KS].

1.2 A Technical Difficulty in the Algorithm

The main task required by the algorithm is thus the efficient computation of all critical orientations. As shown in [LS], the number of critical orientations is $O(kn\lambda_6(kn))$, where $\lambda_s(m)$ is the maximum length of a *Davenport-Schinzel sequence* of order s composed of m symbols (see [HS, ASS]).

Our goal is to compute all these orientations in time $O(kn\lambda_6(kn)\log kn)$. The technique of [LS, KS] does not achieve this goal exactly, which is the source of the problem we are about to discuss; instead it constructs a *superset* of all critical orientations. The size of this superset is also $O(kn\lambda_6(kn))$, and each of these orientations has a placement at which B makes simultaneously three obstacle contacts, but for some of these orientations the corresponding triple-contact placement might not be free. We do not know of any really efficient way to distinguish between the *valid* orientations, whose associated placements are free and the other *spurious* ones. For simplicity, we refer to all computed orientations as critical. Figure 1 shows placements of B at critical orientations.

The reason why the algorithm in [KS] may generate spurious critical orientations lies in its technique (adapted from [LS]) for computing critical orientations. We sketch the technique briefly. Define a *contact pair* as a pair $O = (W, S)$, where W is an obstacle edge or vertex and S is a vertex or edge of B,

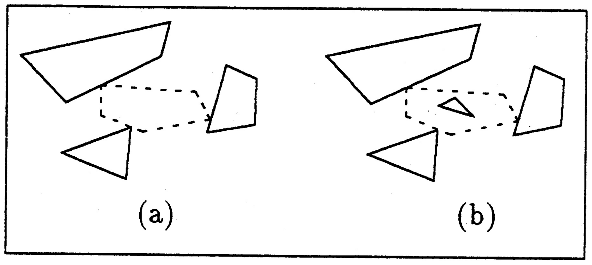

Figure 1: A valid critical orientation (a) and a spurious critical orientation (b); the obstacles are drawn in solid lines and B in dashed lines.

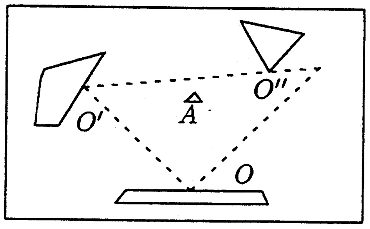

Figure 2: Obstacle containment not encoded in any bounding function

respectively. Given two contact pairs O, O' and an orientation θ, it is shown in [LS], [KS] that if there is a free placement of B at orientation θ where it makes a double contact involving O and O' then one of these pairs, say O', must *bound* the other, in the sense that there is a direction in which we can translate B so that it continues to make the contact O, and during this motion it keeps intersecting the obstacle corner or edge of O', until it reaches the 'end' of the contact O (see the cited papers for more details, see also Figure 2 where O' bounds O, O'' bounds O and O' bounds O'').

This allows us to introduce a collection of *bounding functions* $F_{O,O'}(\theta)$, each measuring the translation distance along O from its starting point to the point at which B (at orientation θ) makes the double contact O, O', where O' bounds O. These are partial functions of θ, and are defined only when such a double contact is possible and when O' bounds O at that contact. It is shown in [LS, KS] that by computing various upper and lower envelopes of subsets of bounding functions, and by a careful interaction of these envelopes, one can obtain (the superset of) all critical orientations.

The main weakness of this technique is that the bounding functions do not represent all the information about collision of B with the obstacles. A typical problematic example is shown in Figure 2; in that figure, as B slides along the contact O it always contains the small obstacle A, and this fact is not encoded in any bounding function, because there exists no double contact (at this orientation) involving O and a contact of B with A. On the other hand, the double contacts O, O' and O, O'' are encoded in their respective bounding functions, hence the critical orientation where B makes the triple contact O, O', O'' is included in the superset of the critical orientations even though it does not represent a free placement of B.

As described above, the algorithm of [KS] constructs the edge-graph EG by sweeping over the list of critical orientations (both valid and spurious) and by processing the changes that occur in VG_θ at each critical orientation θ. The algorithm has to check whether the current critical orientation θ is valid or spurious, to discard spurious orientations, and to update VG_θ and EG at valid orientations. The problem is to find an efficient technique for checking whether the current orientation is valid or not.

A Partial Solution: The paper [KS] suggests the following approach: suppose a triple contact, involving contact pairs O, O', O'', occurs at a critical orientation θ. If the corresponding placement of B is a limit of free placements of B at orientations θ' sufficiently close to, and preceding θ, then θ is clearly valid. Moreover, this can be checked in logarithmic time. However, when this is not the case, i.e. when the critical orientation θ is due to a new component of VG_θ and FP_θ, then this method does not apply, and we have no way to determine, using only local information, whether θ is valid or spurious. This is the problem that we consider in this note.

A Costly Solution: One recent solution to this difficulty was proposed by Sharir and Toledo [ST, T], who apply the algorithm of [KS] to solve extremal polygon containment problems. They prepare several data structures for triangle range searching and for segment intersection queries, and query these structures with each critical placement of B—the placement is free if and only if, at this placement, no vertex of B lies inside an obstacle, B does not contain any obstacle vertex, and no edge of B intersects any obstacle edge. All these tests can be done in a total of $O(k \log kn)$ time per placement, allowing sufficient storage for the data structures (which is available anyway in the algorithm of [KS]). This method runs in time $O(k^2 n \lambda_6(kn) \log kn)$ which is a factor of k worse than the original time bound in [KS]. Our goal in this note is to find an alternative way of overcoming the problem while remaining within the original time bound $O(kn\lambda_6(kn) \log kn)$ of [KS].

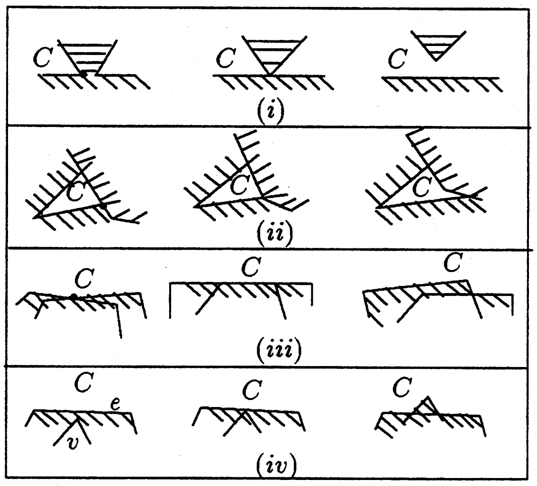

Figure 3: The ways in which C can become non-convex; case (iv) is impossible

2 Our Solution

The task at hand is to find all valid critical orientations at which a new component of FP_θ emerges. Once we know them, we will be able to apply the algorithm of [KS], which computes FP and EG incrementally, and to discard in the process any spurious critical orientation which is supposedly due to an appearance of a new component of FP. Our solution suffers from the potential weakness that at the end of the construction, we do not necessarily have the description of all the cells (connected components) of FP, but we are guaranteed that the cell of FP which contains the initial placement of the robot is fully constructed, and this suffices for most practical applications.

Let θ_1 be an orientation where a new component C emerges. Assuming that the obstacles and B lie in *general position* (see [KS]), C starts as a point at FP_{θ_1} and grows to a small triangle at cross-sections FP_θ for orientations θ slightly larger than θ_1. As θ increases, more edges can be added to C at further critical orientations. Our main observation is that if, during the process of the incremental construction of FP, C is going to be connected to another connected component of FP_{θ^*}, at some orientation θ^*, then at orientations θ greater than but sufficiently close to θ^*, the boundary of C is non-convex. See, for example, Figure 3(i). A cell C whose cross-section remains convex for all θ is therefore called a *dull* cell of FP, as in [AS].

Let θ^* be the smallest orientation greater than θ_1 so that slightly after θ^* the component C is no longer convex. Of course, θ^* might not exist, but then either C is dull, or θ has reached the starting orientation θ_0 (which is also the orientation at which the sweep in θ ends). In the former case, we can ignore C since it is disconnected from the component containing the initial placement of B. The latter case will be treated below in much the same way as the case when θ^* exists. C can become non-convex in one of the following three cases:

(i) A vertex of C becomes coincident with a convex vertex of an expanded obstacle, which then pulls away from an edge of C, connects C to another component and becomes a reflex vertex of the merged component, see Figure 3(i).

(ii) A vertex of C becomes coincident with a convex vertex of an expanded obstacle, which then pulls away from one of the adjacent edges of the vertex of C, exposing the convex corner as a reflex corner of C, see Figure 3(ii).

(iii) Two adjacent edges of C become collinear and then continue to 'bend' outwards, exposing an endpoint of one of them as a reflex corner of C, see Figure 3(iii).

These are the only possible cases in which C can become non-convex, because it must contain a vertex v of some expanded obstacle, reaching the boundary of C from outside C. A simple case analysis shows that there is only one more way in which this can happen: at θ^* the vertex v lies on an edge e of C, and later v pierces through e into C, in the manner shown in Figure 3(iv). However, we claim (without proof—for lack of space) that this case is impossible.

2.1 Computing the Critical Orientations of Concavity

In any of the three possible cases described above, if θ^* denotes the critical orientation at which the corresponding critical configuration arises, then B must make at θ^* two obstacle contacts, one of which involves contact of a corner of B at an obstacle corner (which induces the expanded obstacle corner 'piercing' into C). There are $O(kn)$ contact pairs of this kind, and we can process each of them, in time $O(kn \log kn)$, to find all the corresponding free critical placements of B. To do so, we note that such a corner-corner contact leaves only one degree of freedom for B, namely rotation about the point of contact. For each corner or edge S of B and each convex obstacle A_i we compute the set of orientations at which S meets A_i (while B rotates about the fixed corner-corner contact). Similarly, for each corner or edge W of any obstacle we compute the set of orientations at which B meets W (while maintaining the corner-corner contact). It is easily checked that these sets consist of a total of $O(kn)$ angular intervals, and that they can all be computed in time $O(kn)$. We now compute the union of these 'forbidden' sets, using a simple sorting process that takes $O(kn \log kn)$ time, and the endpoints of the angular intervals forming the union are precisely the (valid!) critical orientations that we seek. Repeating this step for each corner-corner contact, the total running time of this step is $O((kn)^2 \log kn)$. Note that here we can ascertain that all computed orientations and placements are valid. We refer to these orientations as *critical orientations of concavity.*

Remark: Note that, of the three possible cases of critical orientations of concavity, only in case (i) C can merge with another connected component of FP_{θ^*}, so only these orientations are of real interest for us, because only at them C can become a portion of the connected component of FP that contains the initial placement of B. However, computing all the other kinds of critical orientations of concavity does not increase the asymptotic complexity of the algorithm, and makes the above procedure simpler to implement.

2.2 Tracing Events of Concavity Backwards in θ Along FP

Let θ^* be a critical orientation of concavity corresponding to a component C becoming non-convex, as above. Let v be a specific vertex of C at FP_θ, for orientations θ slightly smaller than θ^*, defined as follows: in cases (i) and (ii) we take v to be the vertex of C that is about to become coincident with the convex vertex of the corresponding expanded obstacle; in case (iii) we take v to be the point of intersection of the two adjacent edges of C that are about to become collinear. See Figure 3 where these vertices are highlighted. Note that in each of these cases v is readily determined from the local information about the critical configuration arising at θ^*.

Imagine now that we reverse the direction of the sweep in θ, over the list of critical orientations, starting at θ^* and proceeding in the list backwards until C shrinks to a point and disappears. As this sweep begins, v traces an edge of FP, which may end at a vertex w of FP (at some critical orientation), but, as follows from the structure of C, w must be adjacent to at least one new edge of FP emerging from it in the direction of decreasing θ, and this path (or paths) continue along the boundary of FP until reaching the critical placement where C vanishes.

In other words, if we sweep in θ backwards, tracing these edges of FP and updating them at the appropriate critical orientations, we eventually identify all valid critical orientations at which new components of FP emerge, with the exclusion of dull components that are unreachable from the initial placement of B and have only convex θ-cross-sections.

The backwards sweep in θ can be implemented in the following manner, similar to that used in [KS].

For each contact pair O we maintain a sorted list $L_O = L_O(\theta)$ of contact pairs, so that for each $O' \in L_O(\theta)$ the two contacts O and O' induce a vertex of FP_θ (or, equivalently, induce an edge of FP that crosses the section FP_θ). Not all such vertices will be maintained, but those maintained are guaranteed not to be spurious. The lists $L_O(\theta)$ are maintained as balanced search trees so that searches and updates in any of them can be performed in $O(\log kn)$ time. We initialize these lists by computing the whole cross section FP_{θ_0} at the initial orientation θ_0, as in [KS]. We then sweep from θ_0 backwards through the sorted list Ξ of all critical orientations.

If the current orientation $\theta^* \in \Xi$ is a (valid) critical orientation of concavity, we find the corresponding new vertex v of the corresponding component C of FP_{θ^*}, and the two contact pairs O, O' that induce v. We add O' to L_O and O to $L_{O'}$. (In case (i) we will add both vertices that 'emerge' from the critical placement, because C might lie on either side of this placement.)

Suppose that θ^* is some other critical orientation, corresponding to a simultaneous triple contact involving the contact pairs O, O', O''. We search with O' and O'' in L_O, with O and O'' in $L_{O'}$, and with O and O' in $L_{O''}$. If none of these searches is successful, we ignore θ^*, as it does not affect the structure of the lists we are maintaining, and might as well be spurious. Suppose then that some of these searches succeed, say we find O' in L_O (and, symmetrically, also O in $L_{O'}$). The important observation is that θ^* must then be a valid critical orientation, because the critical placement it represents is a limit placement of free placements of B at orientations $\theta \downarrow \theta^*$. We thus determine the local changes in FP_θ that occur around the corresponding critical placement, and update the relevant lists L_O, $L_{O'}$, $L_{O''}$ as needed. These updates may consist of insertions, deletions, or replacements; they can be directly inferred from the local geometric information describing the critical placement, and ensure that the lists L continue to contain only valid entries after these modifications.

If the local change at θ^* involves a small triangular component of FP_θ that shrinks at θ^* to a point and then vanishes, we add θ^* to an output list Ξ'. We proceed with this sweep until all critical orientations are exhausted. The final output Ξ' gives us all valid critical orientations at which (in the forward sweep in θ) new components of FP_θ, which are not dull, emerge. The correctness of this procedure follows from the invariant, easy to establish by induction, that (a) the procedure maintains only valid contacts in the lists L_O; and (b) the procedure traces all paths π which consist of a sequence of edges of FP, are monotonically decreasing in θ, and start at a designated vertex v of some critical orientation of concavity. This invariant, combined with the observation made above concerning the backward θ-sweep of convex cross sections C, imply the asserted correctness.

2.3 Computing FP

After identifying in this manner all valid critical placements where new (non-dull) components truly emerge, we now apply the original algorithm of [KS], with the following modification. When we reach a critical orientation θ^*, we check if it is an orientation where a new component of FP_θ (potentially) emerges. If so, we check if θ^* belongs to Ξ'. If not, we ignore it because it must be spurious or involve a dull component. If θ^* does belong to Ξ', we insert the new resulting vertices of FP_θ into the appropriate lists of contact pairs. If θ^* does not involve a newly emerging component, and is defined by three contacts O, O', O'', we search each of these contacts in the lists of the two other contacts, as in the preceding backward-sweeping stage. If none of these searches succeed, we discard θ^*, since it is spurious or corresponds to a dull component. If some of the searches succeed, we know that θ^* is valid, and update the lists as required, using the local data concerning the critical placement that θ^* represents.

This modified algorithm correctly traces all the edges of non-dull components of FP. Indeed, the algorithm maintains the invariant that, at any orientation θ, its data structures represent all true vertices of FP_θ that lie in non-dull components, and only those vertices. We omit a full proof of this fact, which is a consequence of the analysis given in [KS] and of the discussion given above.

To summarize, we propose to use the following 4-stage algorithm:
(a) Compute (a superset of) all critical orientations, as in [LS, KS].
(b) Compute all critical orientations of concavity.
(c) Perform the backwards sweep in θ, to obtain the list of all valid critical orientations where new components of FP_θ emerge.
(d) Finally apply the algorithm of [KS], modified as above.

Thus the overall running time of the algorithm is $O(kn\lambda_6(kn)\log kn)$, and is thus as efficient asymptotically as the original algorithm of [KS].

References

[ASS] P. Agarwal, M. Sharir and P. Shor, Improved bounds for the length of general Davenport Schinzel sequences, *J. Combin. Theory, Ser. A* 52 (1989), 228–274.

[AS] B. Aronov and M. Sharir, Triangles in space, or building (and analyzing) castles in the air, *Combinatorica* 10 (1990), 137–173.

[GSS] L.J. Guibas, M. Sharir and S. Sifrony, On the general motion planning problem with two degrees of freedom, *Discrete Comput. Geom.* 4 (1989), 491–521.

[HS] S. Hart and M. Sharir, Nonlinearity of Davenport-Schinzel sequences and of generalized path compression schemes, *Combinatorica* 6 (1986), 151–177.

[KLPS] K. Kedem, R. Livne, J. Pach and M. Sharir, On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles, *Discrete Comput. Geom.* 1 (1986), 59–72.

[KS] K. Kedem and M. Sharir, An efficient motion-planning algorithm for a convex polygonal object in two-dimensional polygonal space, *Discrete Comput. Geom.* 5 (1990), 43–75.

[LS] D. Leven and M. Sharir, On the number of critical free contacts of a convex polygonal object moving in 2-D polygonal space, *Discrete Comput. Geom.* 2 (1987), 255–270.

[ST] M. Sharir and S. Toledo, Extremal polygon containment problems, Manuscript, 1991.

[T] S. Toledo, Extremal polygon containment problems, *Proc. 7th Ann. Symp. on Computational Geometry,* 1991, 176–185.